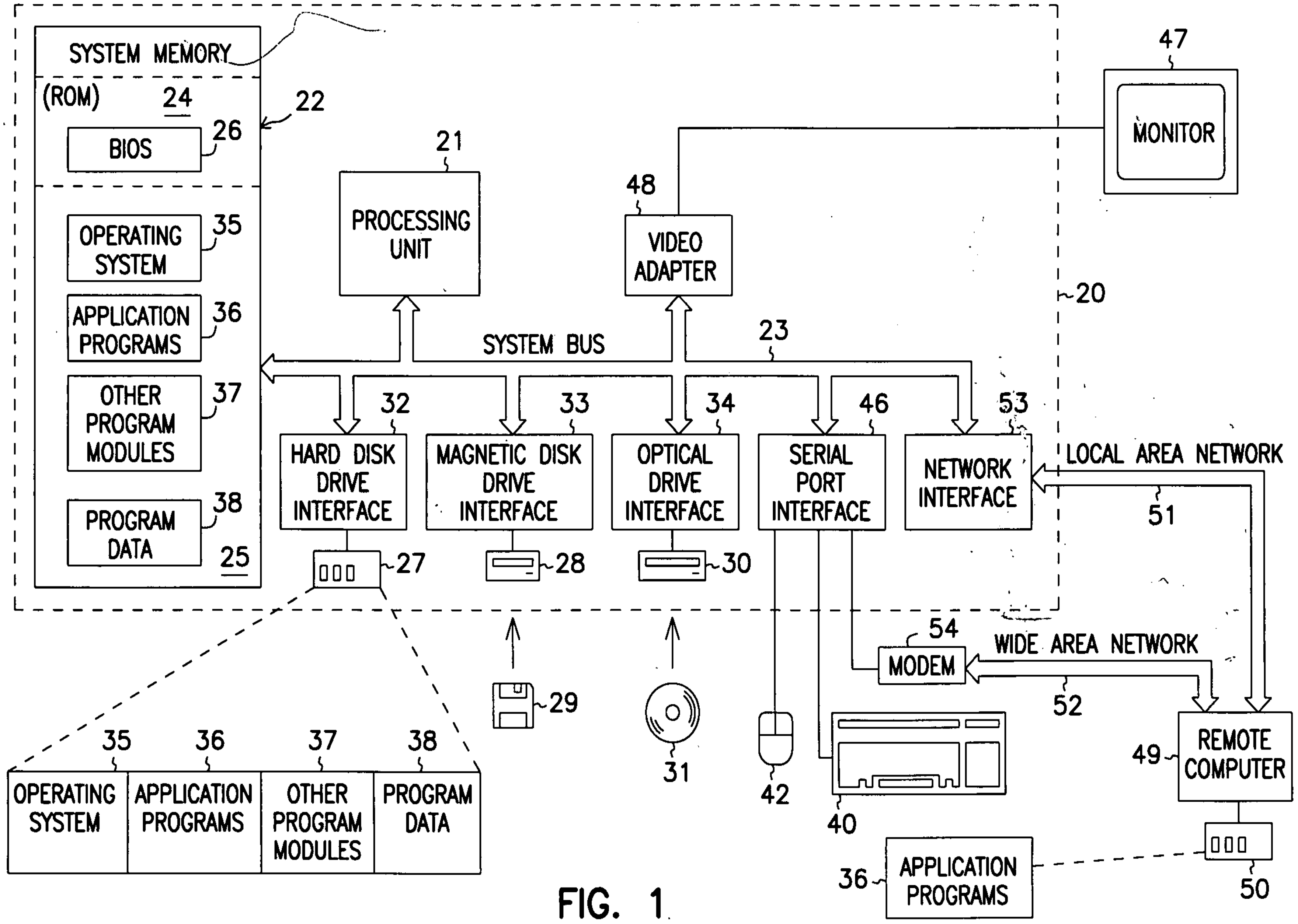

SYSTEM MEMORY
(ROM)
24
BIOS
26
22
OPERATING SYSTEM
35
APPLICATION PROGRAMS
36
OTHER PROGRAM MODULES
37
PROGRAM DATA
38
25
PROCESSING UNIT
21
VIDEO ADAPTER
48
MONITOR
47
20
SYSTEM BUS
23
HARD DISK DRIVE INTERFACE
32
MAGNETIC DISK DRIVE INTERFACE
33
OPTICAL DRIVE INTERFACE
34
SERIAL PORT INTERFACE
46
NETWORK INTERFACE
53
LOCAL AREA NETWORK
51
27
28
30
29
31
42
40
MODEM
54
WIDE AREA NETWORK
52
REMOTE COMPUTER
49
50
APPLICATION PROGRAMS
36
OPERATING SYSTEM
35
APPLICATION PROGRAMS
36
OTHER PROGRAM MODULES
37
PROGRAM DATA
38

FIG. 1

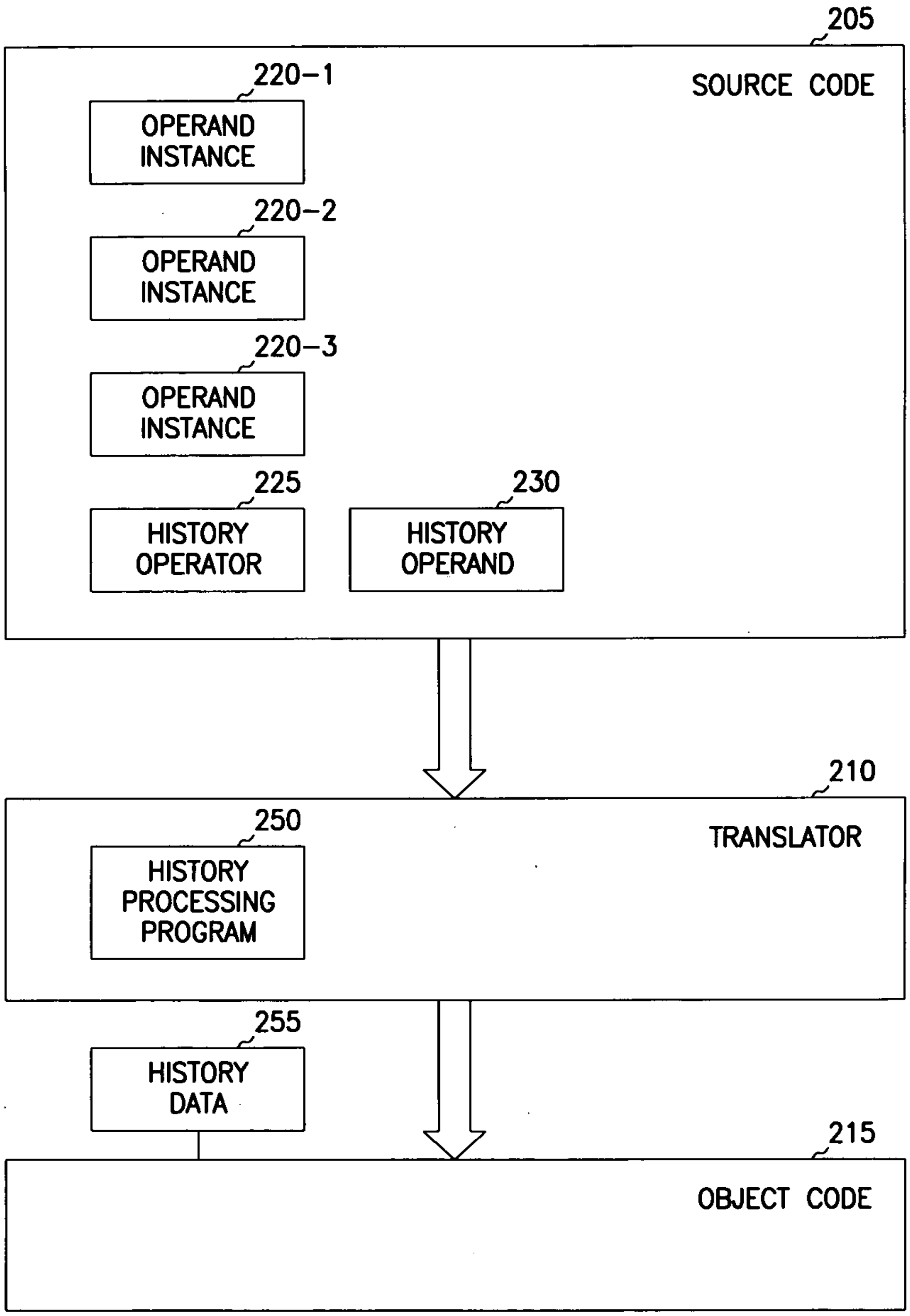

205
SOURCE CODE
220-1
OPERAND INSTANCE
220-2
OPERAND INSTANCE
220-3
OPERAND INSTANCE
225
HISTORY OPERATOR
230
HISTORY OPERAND
210
TRANSLATOR
250
HISTORY PROCESSING PROGRAM
255
HISTORY DATA
215
OBJECT CODE

FIG. 2

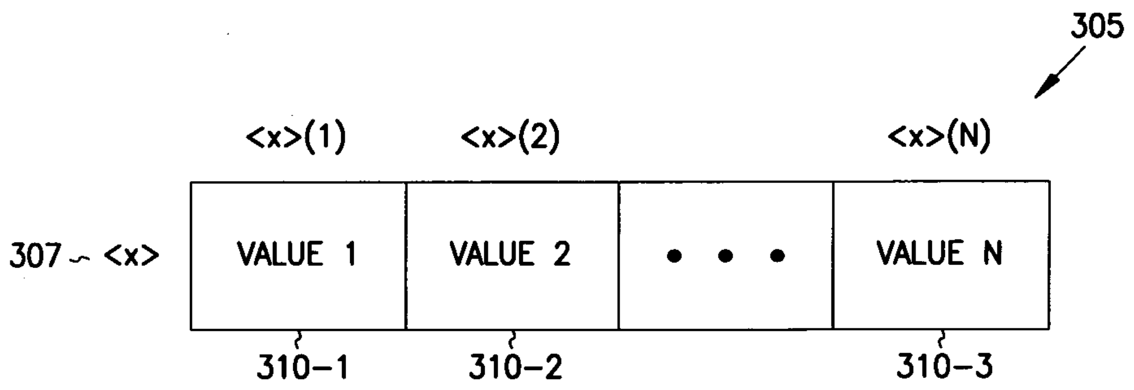

305
<x>(1)
<x>(2)
<x>(N)
307
<x>
VALUE 1
VALUE 2
VALUE N
310-1
310-2
310-3

FIG. 3A

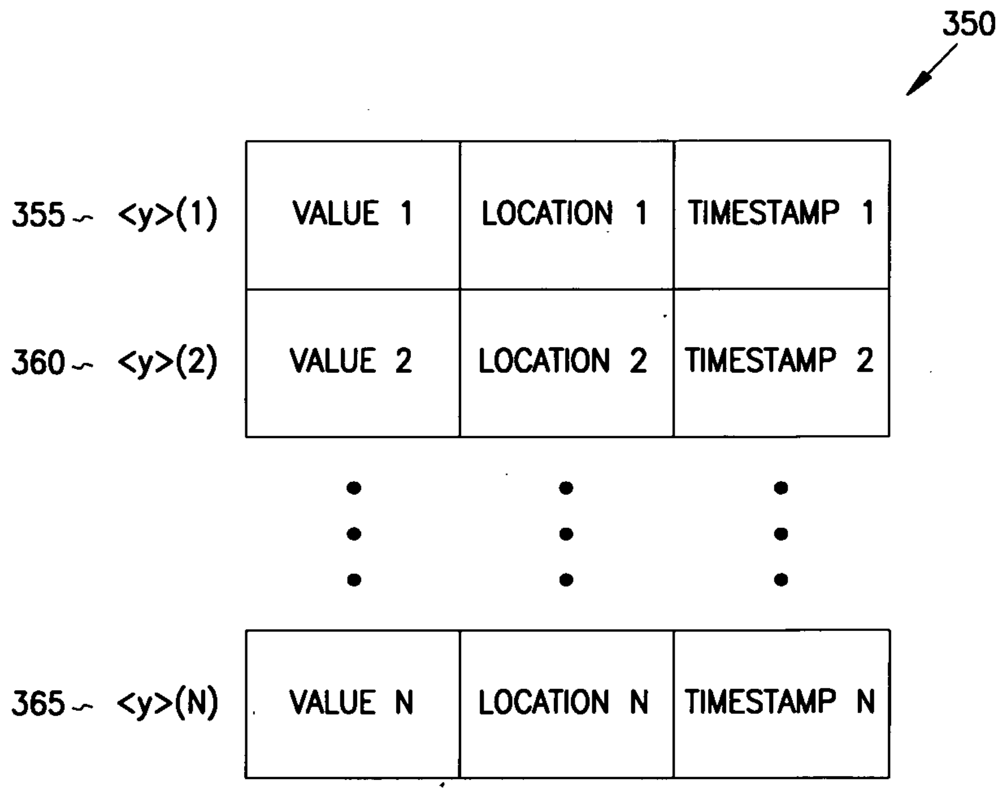
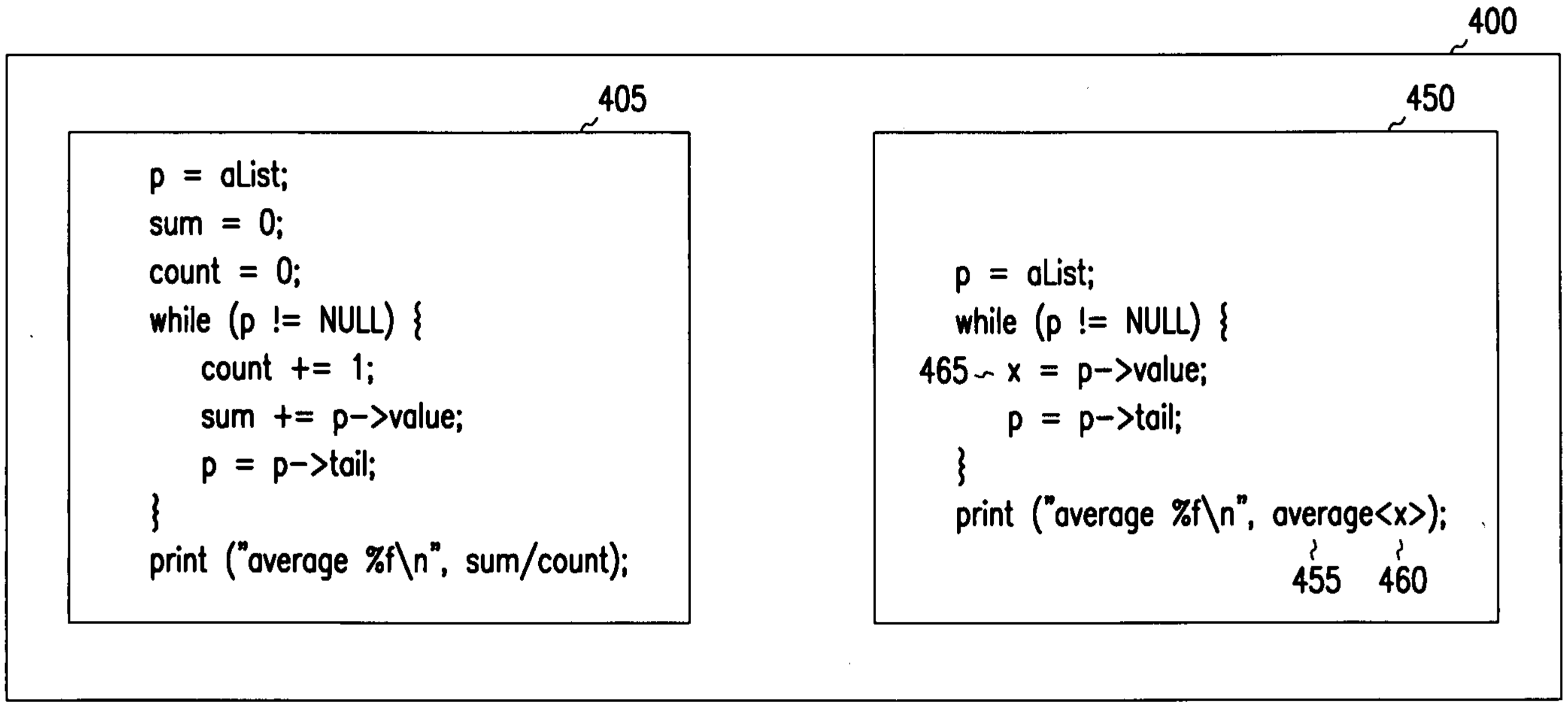

350
355
<y>(1)
VALUE 1
LOCATION 1
TIMESTAMP 1
360
<y>(2)
VALUE 2
LOCATION 2
TIMESTAMP 2
365
<y>(N)
VALUE N
LOCATION N
TIMESTAMP N

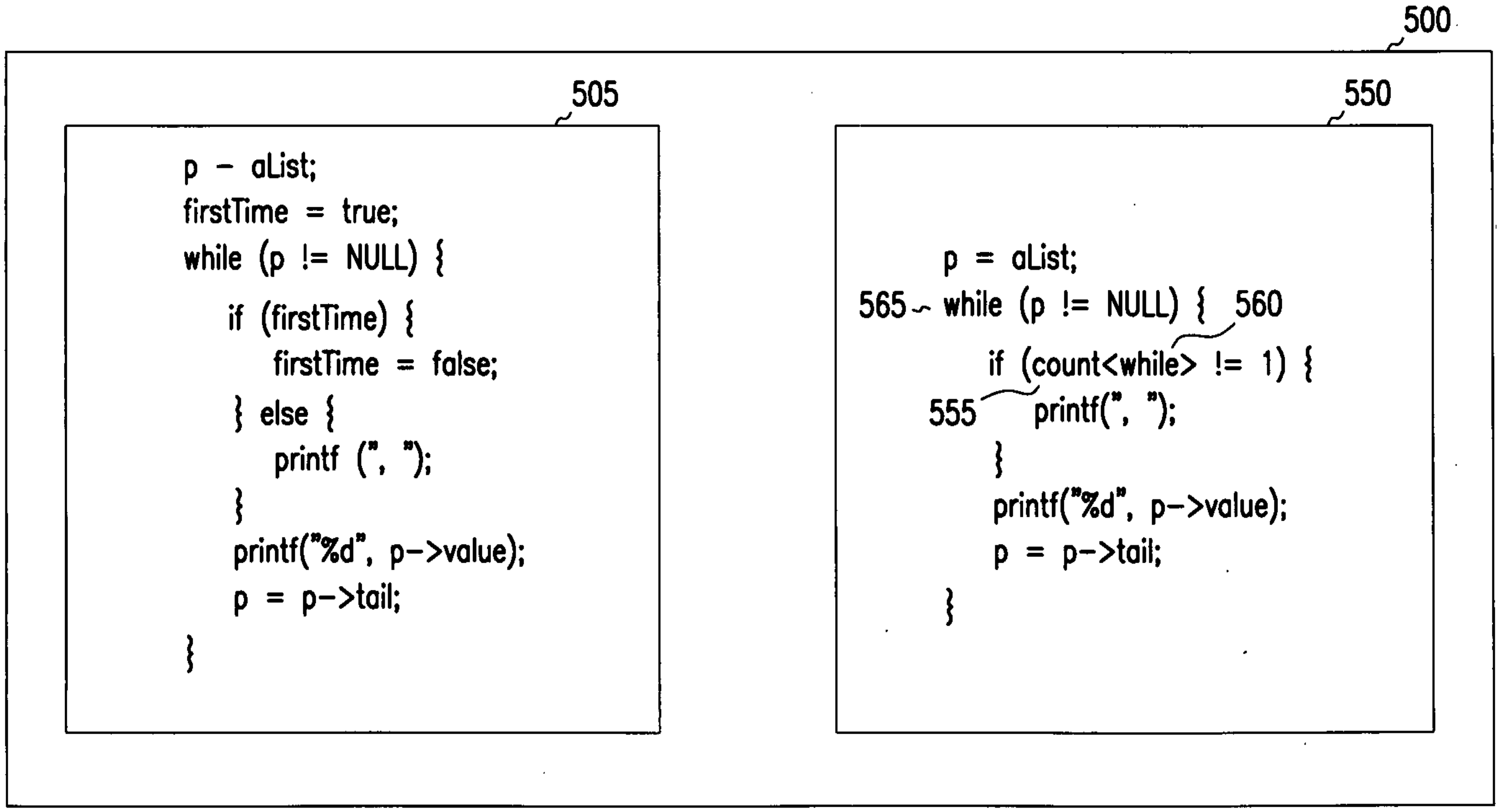
FIG. 3B



```
p = aList;
sum = 0;
count = 0;
while (p != NULL) {
    count += 1;
    sum += p->value;
    p = p->tail;
}
print ("average %f\n", sum/count);
```

```
p = aList;
while (p != NULL) {
    x = p->value;
    p = p->tail;
}
print ("average %f\n", average<x>);
```

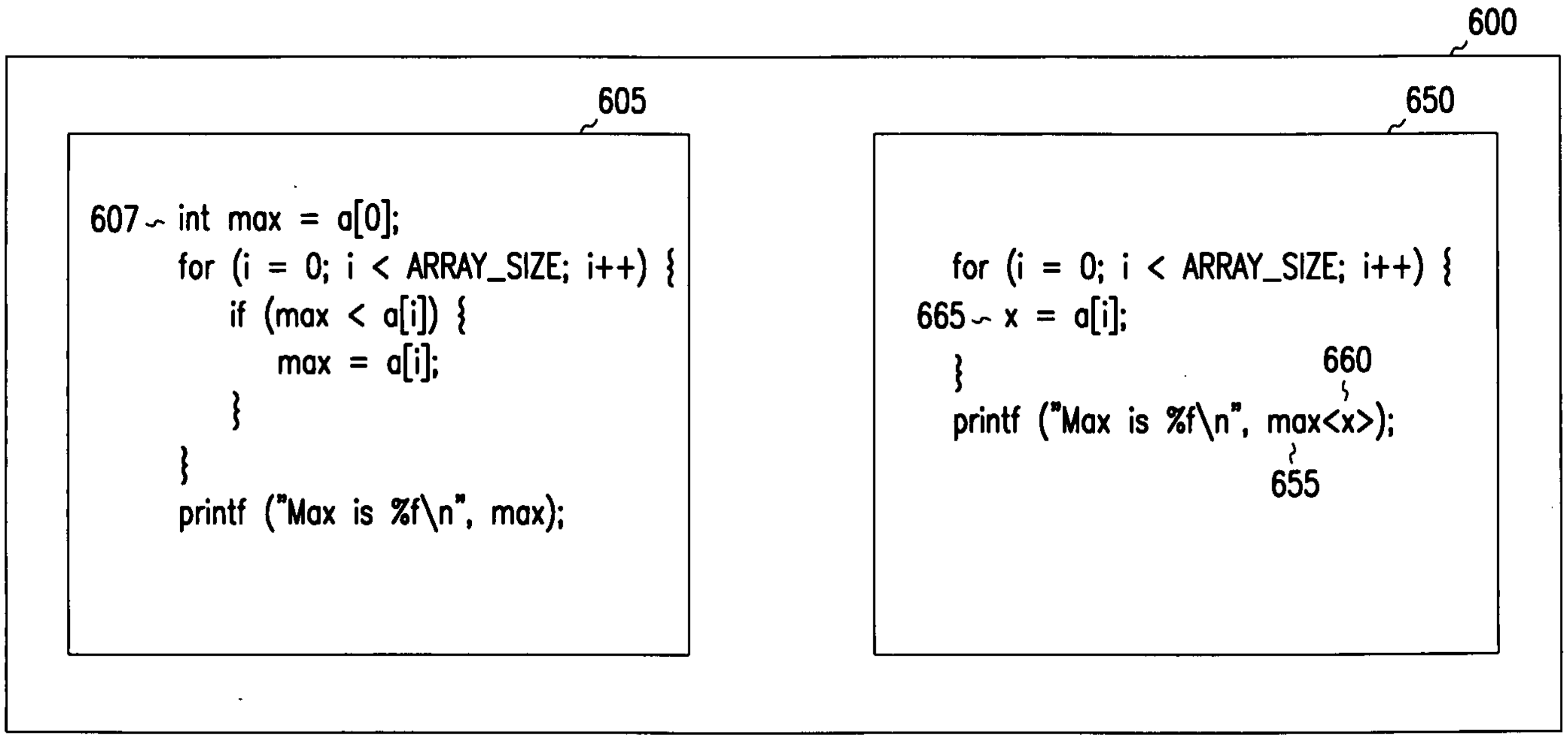
FIG. 4



```
p - aList;
firstTime = true;
while (p != NULL) {
    if (firstTime) {
        firstTime = false;
    } else {
        printf (", ");
    }
    printf("%d", p->value);
    p = p->tail;
}
```

```
p = aList;
while (p != NULL) {
    if (count<while> != 1) {
        printf(", ");
    }
    printf("%d", p->value);
    p = p->tail;
}
```

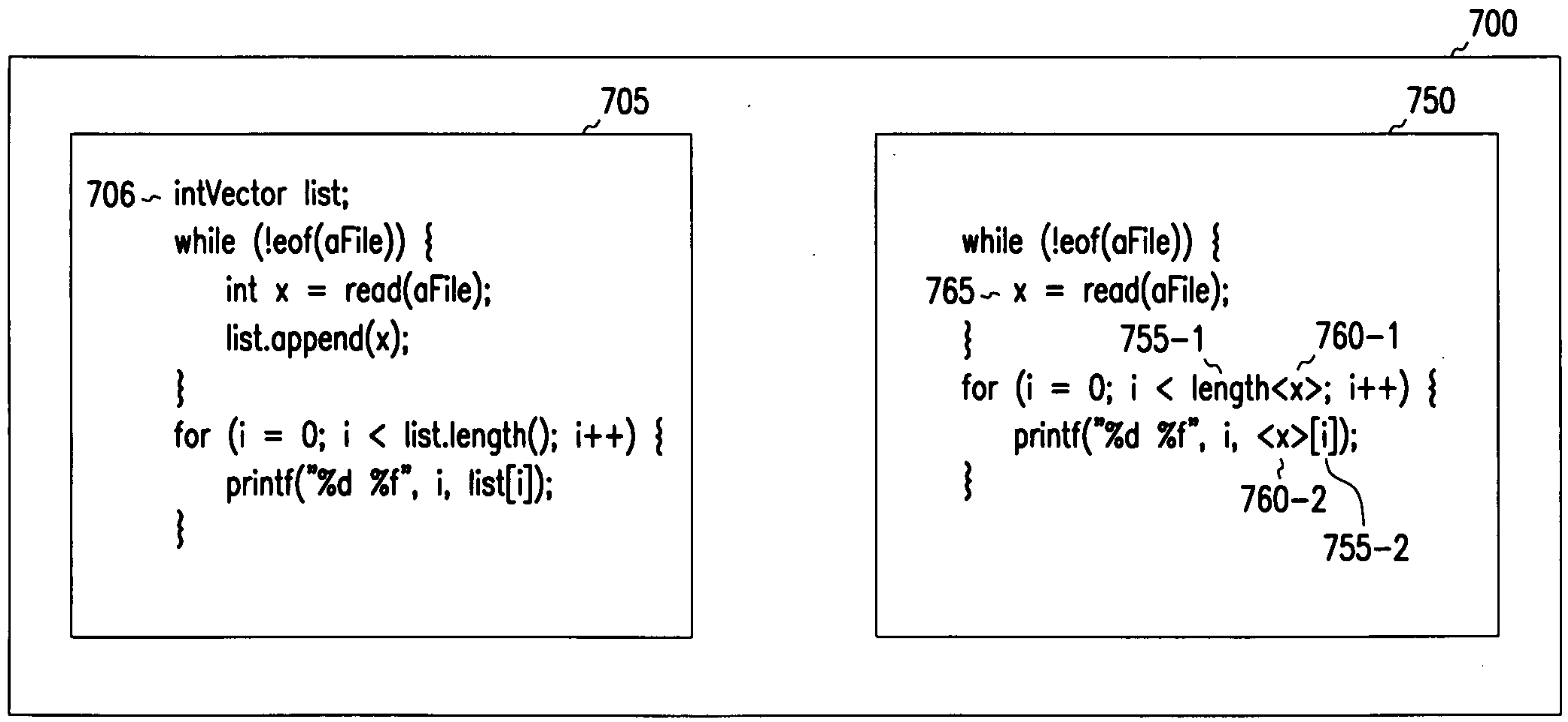
FIG. 5



```
607~ int max = a[0];
     for (i = 0; i < ARRAY_SIZE; i++) {
         if (max < a[i]) {
             max = a[i];
         }
     }
     printf ("Max is %f\n", max);
```

```
  for (i = 0; i < ARRAY_SIZE; i++) {
665~ x = a[i];
  }
  printf ("Max is %f\n", max<x>);
```

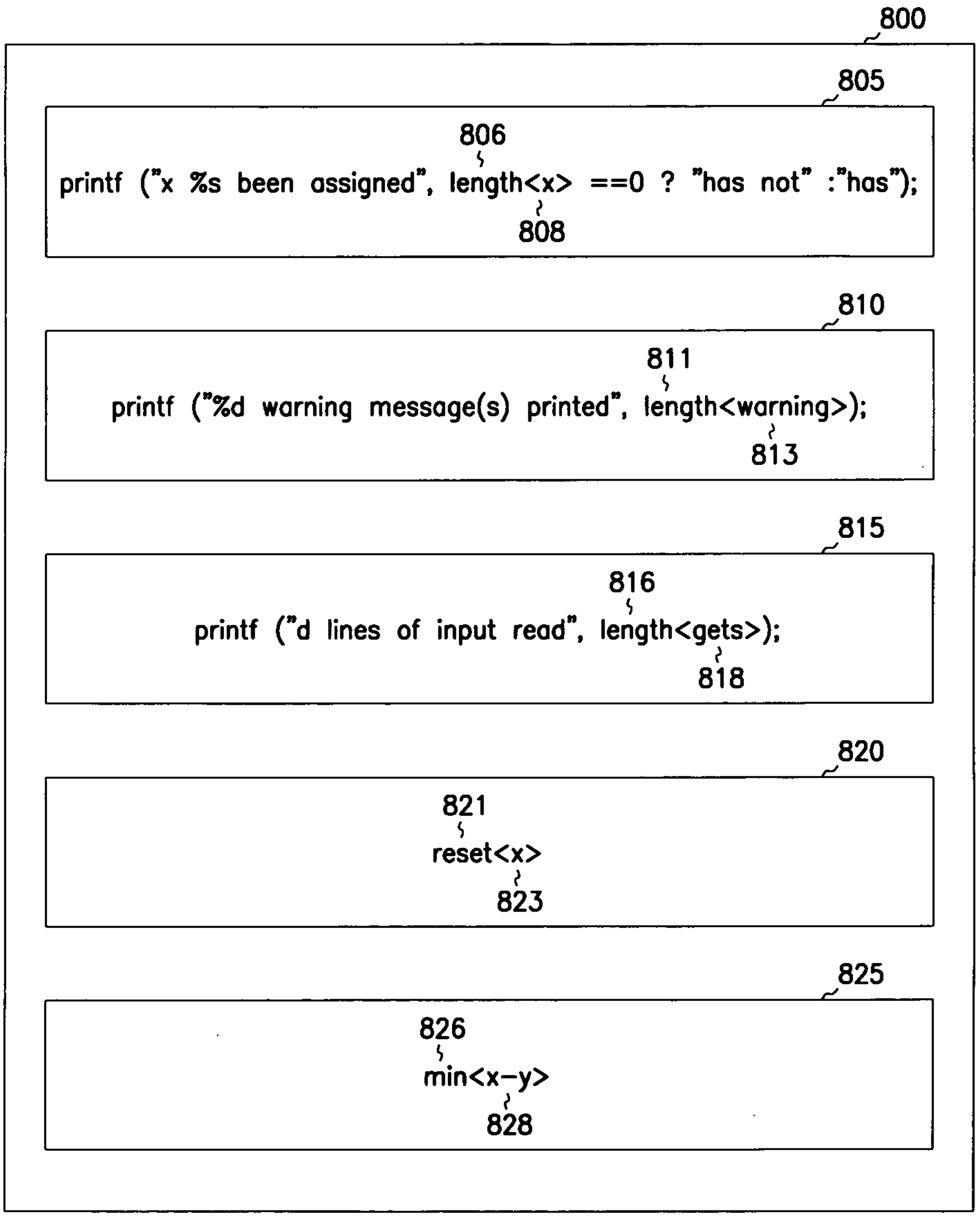
FIG. 6



```
intVector list;
    while (!eof(aFile)) {
        int x = read(aFile);
        list.append(x);
    }
    for (i = 0; i < list.length(); i++) {
        printf("%d %f", i, list[i]);
    }
```

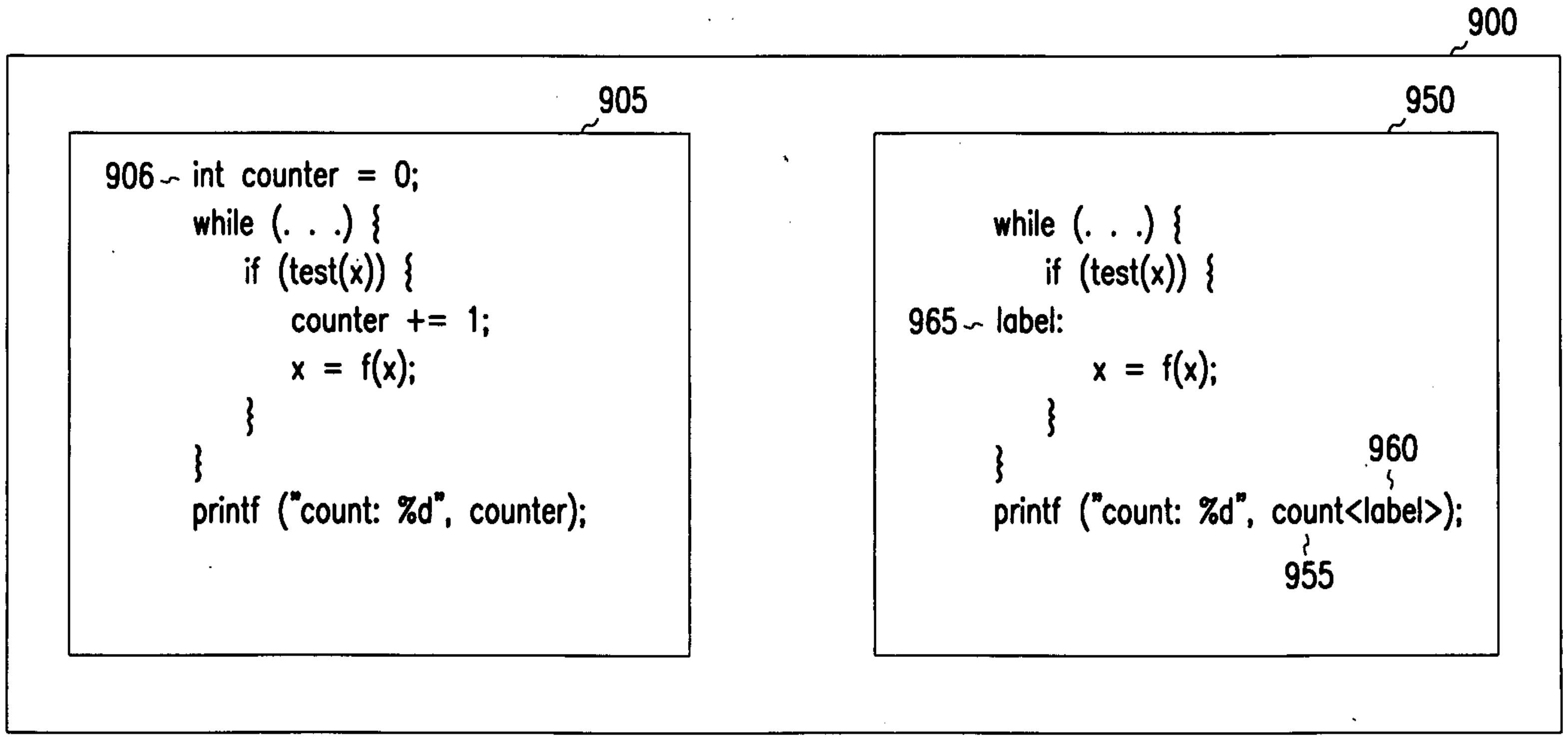
```
  while (!eof(aFile)) {
  x = read(aFile);
  }
  for (i = 0; i < length<x>; i++) {
      printf("%d %f", i, <x>[i]);
  }
```

FIG. 7



```
printf ("x %s been assigned", length<x> ==0 ? "has not" :"has");

printf ("%d warning message(s) printed", length<warning>);

printf ("d lines of input read", length<gets>);

reset<x>

min<x-y>
```

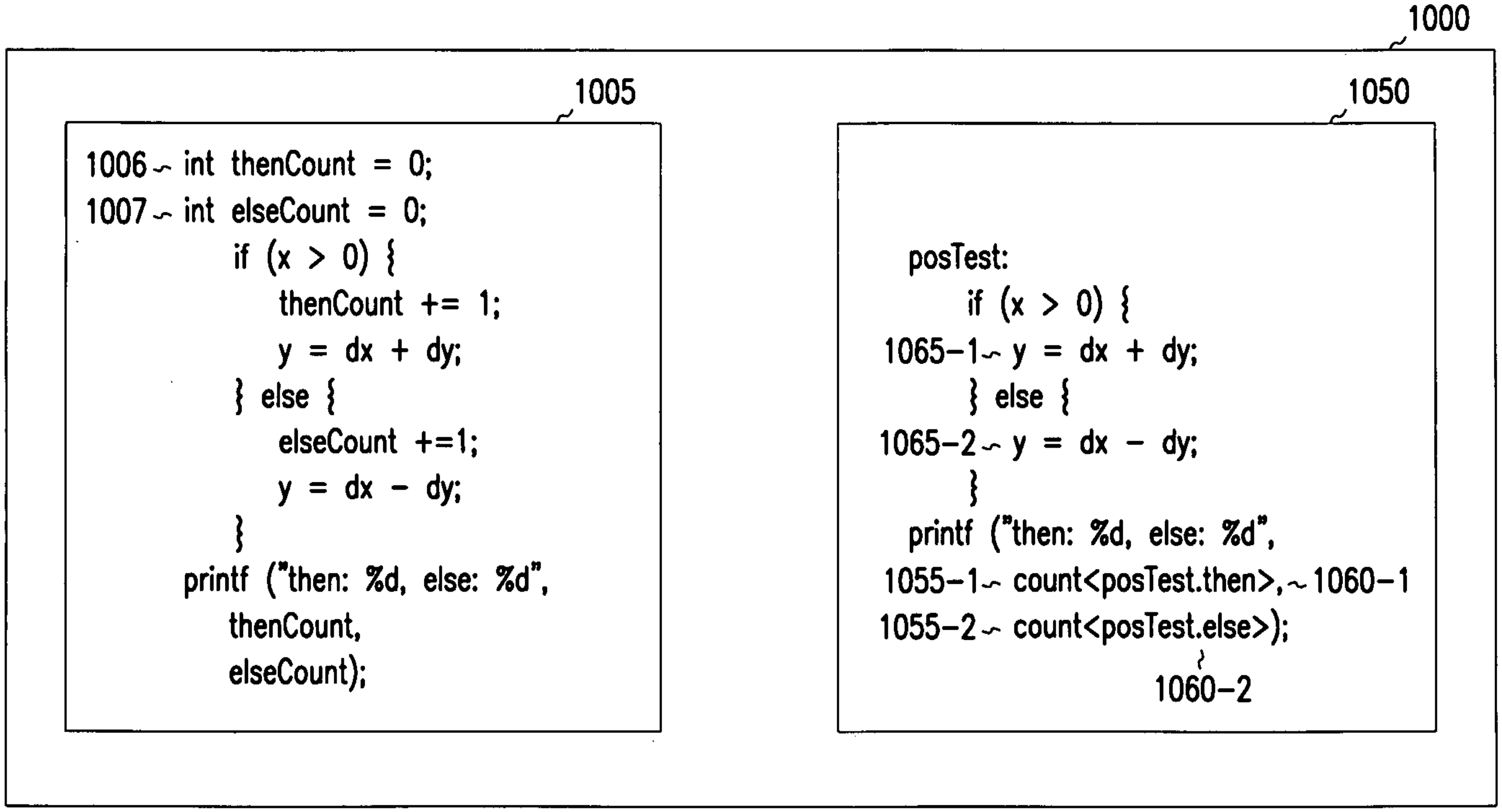
FIG. 8



```
906~ int counter = 0;
     while (. . .) {
         if (test(x)) {
             counter += 1;
             x = f(x);
         }
     }
     printf ("count: %d", counter);
```

```
     while (. . .) {
         if (test(x)) {
965~ label:
             x = f(x);
         }
     }
     printf ("count: %d", count<label>);
```

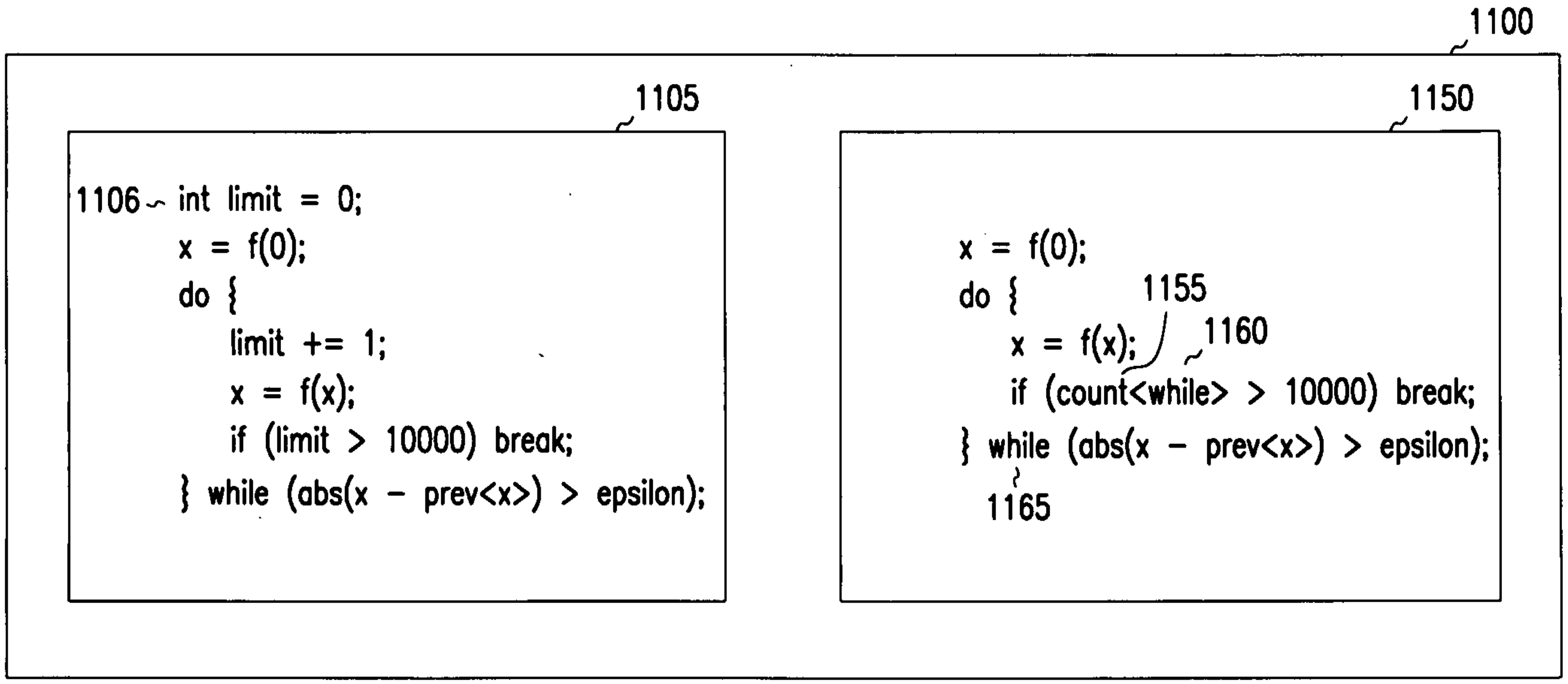
FIG. 9



```
1006~ int thenCount = 0;
1007~ int elseCount = 0;
        if (x > 0) {
            thenCount += 1;
            y = dx + dy;
        } else {
            elseCount +=1;
            y = dx - dy;
        }
    printf ("then: %d, else: %d",
        thenCount,
        elseCount);
```

```
    posTest:
        if (x > 0) {
1065-1~ y = dx + dy;
        } else {
1065-2~ y = dx - dy;
        }
    printf ("then: %d, else: %d",
1055-1~ count<posTest.then>,~1060-1
1055-2~ count<posTest.else>);
                     1060-2
```

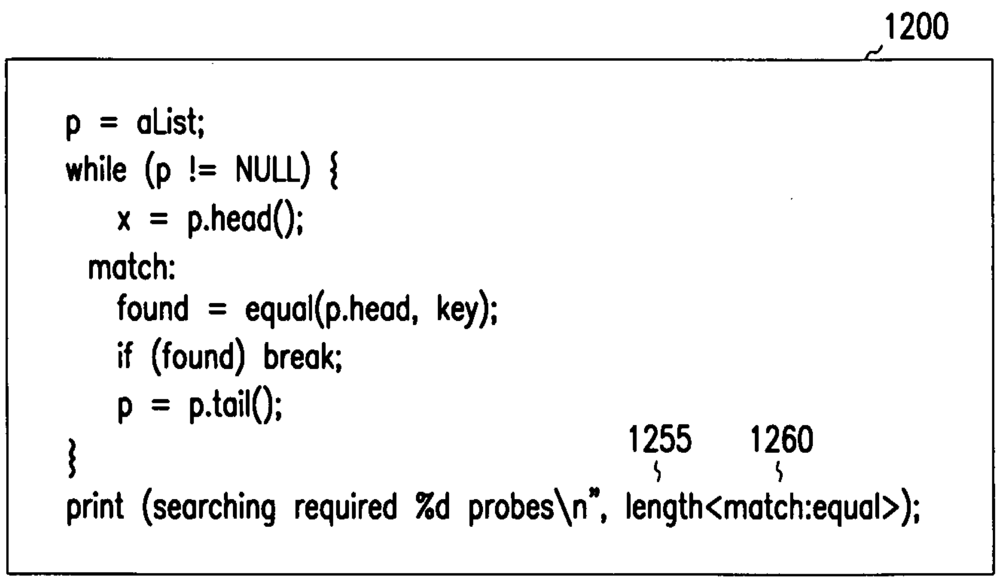
FIG. 10



```
int limit = 0;
    x = f(0);
    do {
        limit += 1;
        x = f(x);
        if (limit > 10000) break;
    } while (abs(x - prev<x>) > epsilon);
```

```
x = f(0);
do {
    x = f(x);
    if (count<while> > 10000) break;
} while (abs(x - prev<x>) > epsilon);
```

FIG. 11


1200
1255
1260

```
p = aList;
while (p != NULL) {
    x = p.head();
  match:
    found = equal(p.head, key);
    if (found) break;
    p = p.tail();
}
print (searching required %d probes\n", length<match:equal>);
```

FIG. 12

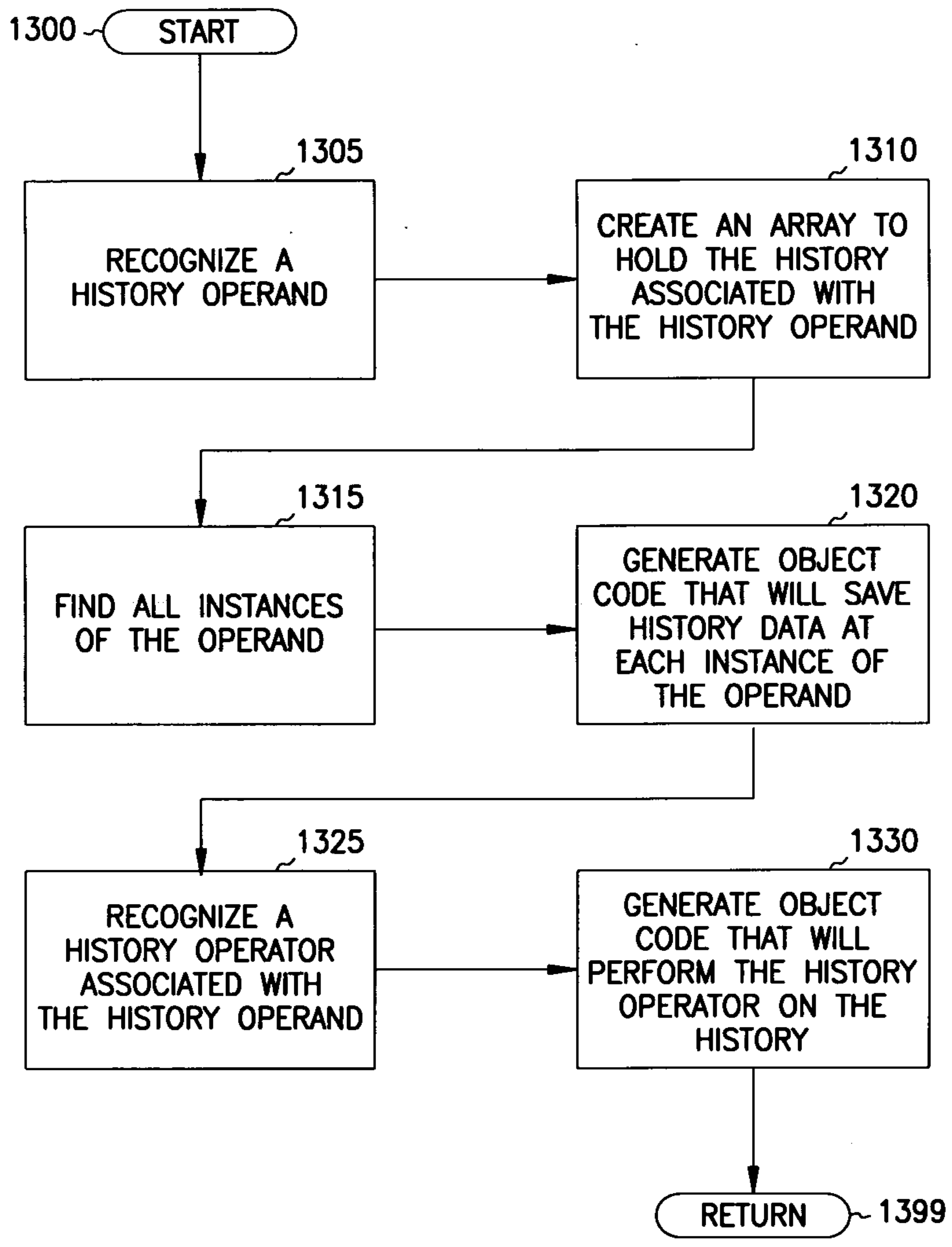

1300
START
1305
RECOGNIZE A HISTORY OPERAND
1310
CREATE AN ARRAY TO HOLD THE HISTORY ASSOCIATED WITH THE HISTORY OPERAND
1315
FIND ALL INSTANCES OF THE OPERAND
1320
GENERATE OBJECT CODE THAT WILL SAVE HISTORY DATA AT EACH INSTANCE OF THE OPERAND
1325
RECOGNIZE A HISTORY OPERATOR ASSOCIATED WITH THE HISTORY OPERAND
1330
GENERATE OBJECT CODE THAT WILL PERFORM THE HISTORY OPERATOR ON THE HISTORY
RETURN
1399

FIG. 13